

## YOLO Weapon detection Project

### INDEX

1. INTRODUCTION
2. LITERATURE SURVEY
3. IDEATION AND PROPOSED SOLUTION
4. REQUIREMENT ANALYSIS
5. PRODUCT DESIGN
6. PROJECT PLANNING AND SCHEDULING
7. CODING AND SOLUTIONING
8. PERFORMANCE TESTING
9. RESULTS
10. ADVANTAGES AND DISADVANTAGES
11. CONCLUSION
12. FUTURE SCOPE
13. APPENDIX

## 1. INTRODUCTION

### Project Overview

A YOLO-based weapon detection project involves using the YOLO (You Only Look Once) algorithm to detect weapons in images or video streams. The YOLO algorithm is a real-time object detection system that is particularly well-suited for detecting multiple objects in an image or video frame. The project involves training a YOLO model on a dataset of images and videos that contain weapons, and then using the trained model to detect weapons in new images or videos.

### Purpose

The project's goal is to provide a tool that can help identify and prevent acts of violence by detecting weapons in real-time, thereby making our neighbourhoods and public spaces safer.

## 2. LITERATURE SURVEY

The paper [1] provides a detailed review on evolution of the YOLO model YOLO (You Only Look Once) framework, from its inception with YOLOv1 to the latest version, YOLOv8.

The paper [2] introduces an approach for detecting pistols in surveillance videos, utilizing the YOLO (You Only Look Once) models, specifically YOLOv5, YOLOv6, and YOLOv8.

The paper [3] summarizes the models of the YOLO series and provides a detailed analysis of the critical feature of each model.

[1]Terven, J. and Cordova-Esparza, D., 2023. A comprehensive review of YOLO: From YOLOv1 to YOLOv8 and beyond. arXiv preprint arXiv:2304.00501.

[2] Doan, T.S., Nguyen, T.K.T. and Vo, T.A., 2023. Weapon Detection with YOLO Model Version 5, 7, 8.

[3]Wang, X., Li, H., Yue, X. and Meng, L., 2023. A comprehensive survey on object detection YOLO. Proceedings <http://ceur-ws.org> ISSN, 1613, p.0073.

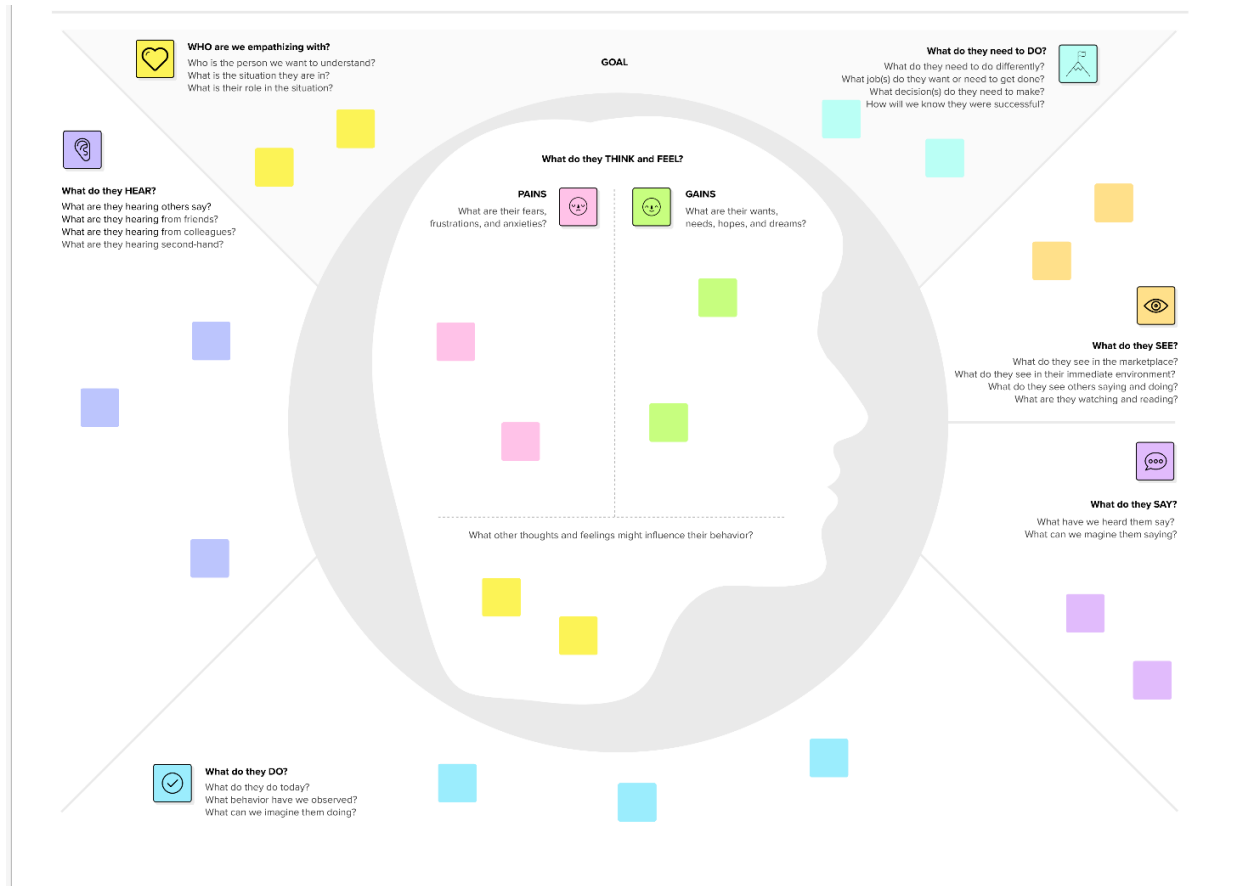
### 3. IDEATION AND PROPOSED SOLUTION

#### Empathy Map Canvas:

An empathy map is a simple, easy-to-digest visual that captures knowledge about a user's behaviors and attitudes.

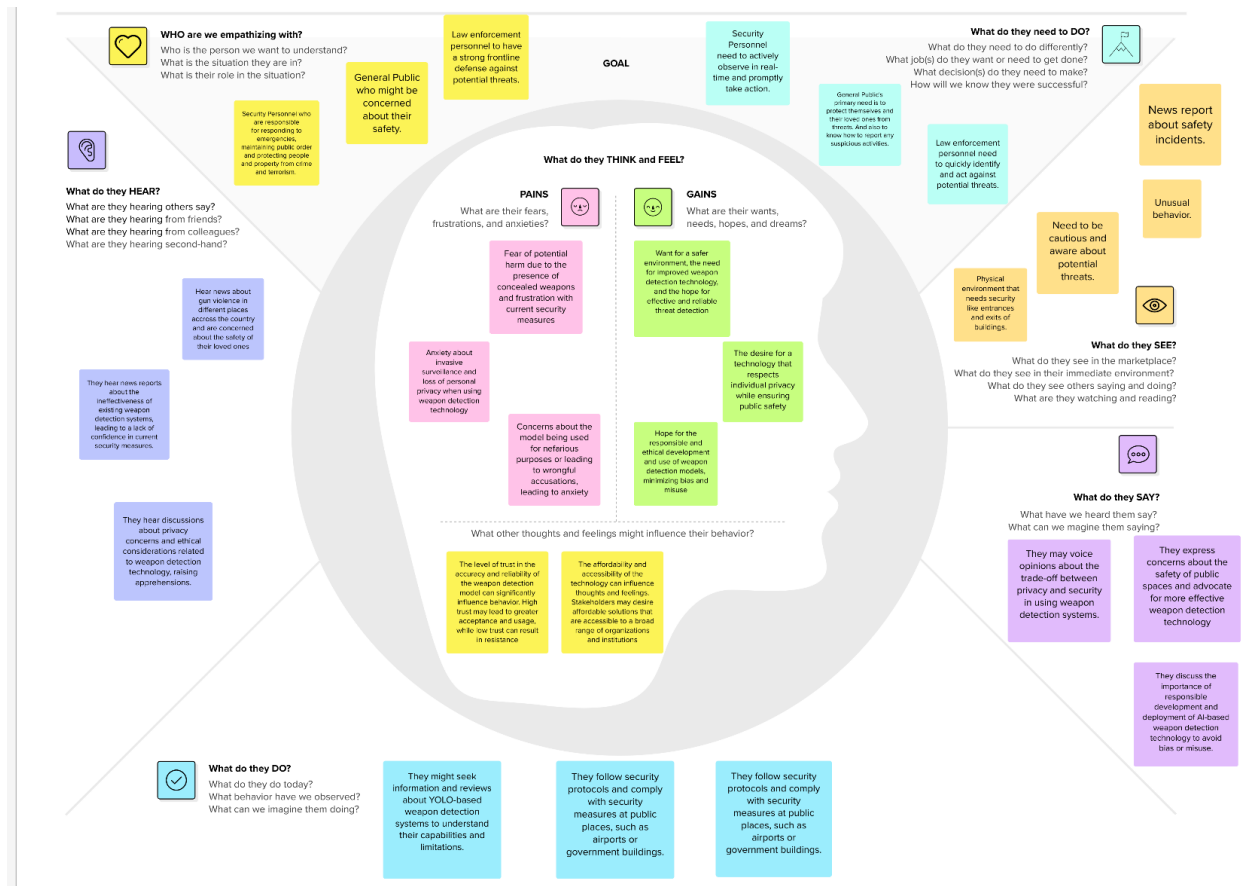
It is a useful tool to help teams better understand their users.

Creating an effective solution requires understanding the true problem and the person who is experiencing it. The exercise of creating the map helps participants consider things from the user's perspective along with his or her goals and challenges.



Reference:(for below map)

<https://app.mural.co/t/guidedprojectyoloweapondetec7977/m/guidedprojectyoloweapondetec7977/1697188386012/2d44636e484193316128f40c8f0c089862718cc2?sender=u4478e7ddff30221d8c126879>



## Arming Against Violence - Yolo-Based Weapon Detection

Reference link:

<https://app.mural.co/t/guidedprojectyoloweapondetec7977/m/guidedprojectyoloweapondetec7977/1697190122933/8ae7716d713cd2ec3a183221f7073df6c365a21a?sender=u4478e7ddff30221d8c126879>

Brainstorming ideas is a creative process where a group generates a list of potential solutions, suggestions, or concepts for a specific problem or project. Voting in brainstorming involves participants selecting and prioritizing their favorite or most promising ideas from the list to determine which ones should be

pursued further.

## Brainstorm

Write down any ideas that come to mind that address your problem statement.

---

### Person 1: Charvi Upreti

Portal to allow general public to report suspicious behavior's.

Method to integrate in existing CCTV architecture to deploy it faster.

Real time alerts based on detected threats.

### Person 2: Girish K

App that uses areas cctv network to detect weapons and warn users to stay away

Portal to allow general public to report suspicious behavior's.

Open source collaboration on YOLO model for usage by everyone and improvements

### Person 3: Sukanth K

Creating systems to ensuring privacy by design in YOLO based weapon detection system

Developing a mobile app that uses the model for weapon detection and can alert users

Bootcamp or classes on self-defense to educate anyone and everyone to defend themselves from weapons and appropriate steps to follow in case of emergency

## Group ideas

Take turns sharing your ideas while clustering similar or related notes as you go. Once all sticky notes have been grouped, give each cluster a sentence-like label. If a cluster is bigger than six sticky notes, try and see if you can break it up into smaller sub-groups.

---

### APP Ideas

Portal to allow general public to report suspicious behavior's.

App that uses areas cctv network to detect weapons using an AI model and warn users to stay away from respective area

Developing a mobile app that uses the model for weapon detection and can alert users

### Real Time Threat detection(CCTV)

Real time alerts based on detected threats.

Method to integrate in existing CCTV architecture to deploy it faster.

App that uses areas cctv network to detect weapons using an AI model and warn users to stay away from respective area

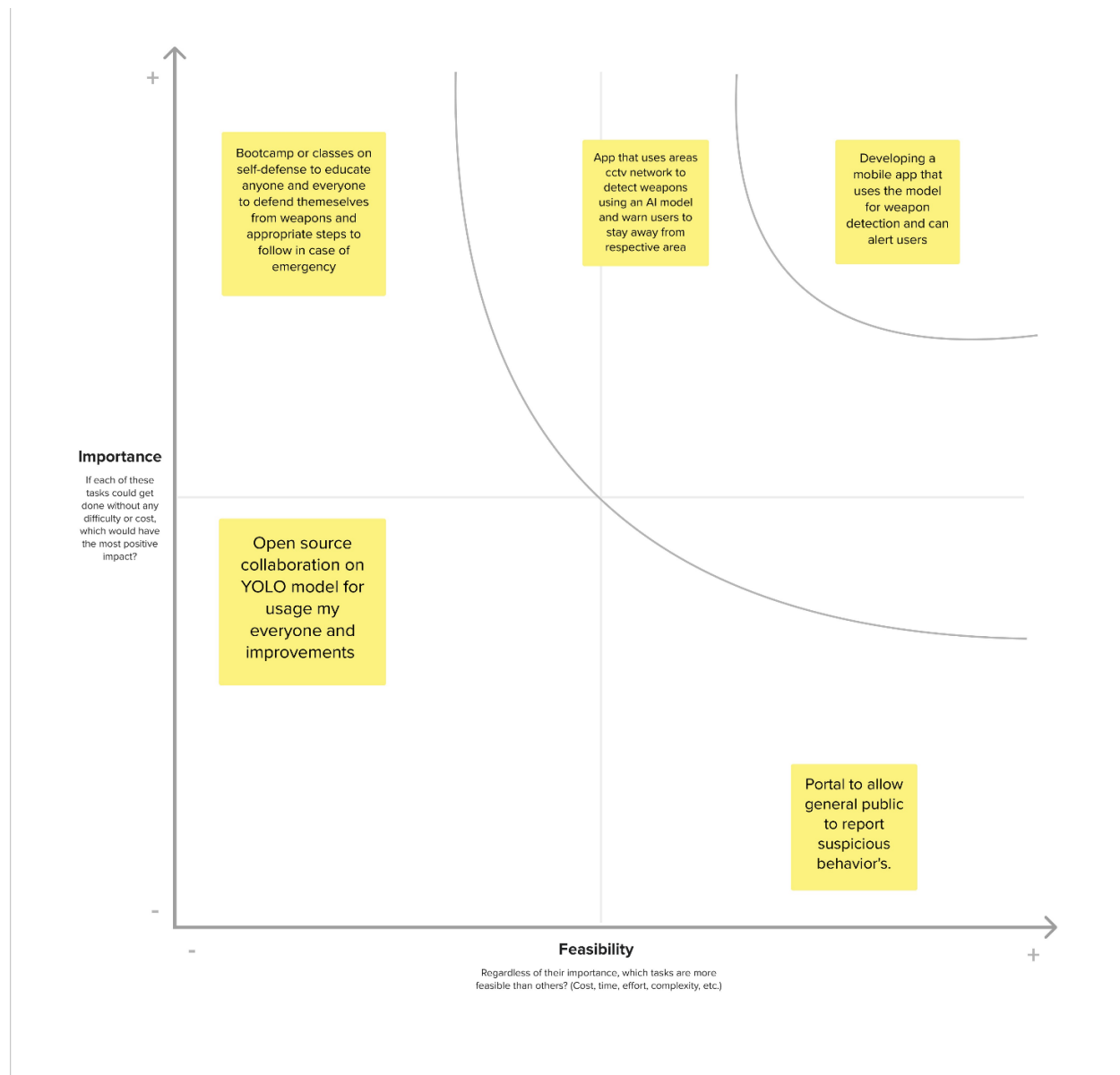
### Other Unique Ideas

Open source collaboration on YOLO model for usage by everyone and improvements

Bootcamp or classes on self-defense to educate anyone and everyone to defend themselves from weapons and appropriate steps to follow in case of emergency

## Idea prioritization

Idea prioritization is the process of ranking or assessing ideas based on specific criteria such as feasibility, impact, cost, or strategic importance to determine which ideas should be implemented or pursued first.



The YOLO weapon detection model and its companion app stand out as the top choice as a team faced with selecting the most significant and practical project among multiple possibilities. In our opinion, this technology offers superior significance and viability compared to the alternative ideas. The YOLO model gives us a powerful weapon to successfully address violence that the alternatives cannot equal thanks to its real-time threat identification, unmatched accuracy, and scalability. The YOLO app also encourages community participation in reporting suspicious activity, which distinguishes it from competing ideas and fosters collaboration and inclusivity.

The YOLO model offers a more focused, flexible, and economical strategy when compared to building a citywide CCTV network report and monitoring system, which can be challenging in terms of cost and complexity. It streamlines the procedure by

making use of current surveillance infrastructure. This choice is the best fit for our team because it addresses the problem statement effectively and efficiently.

#### 4. REQUIREMENT ANALYSIS

##### 1. Problem Statement (Problem to be solved)

To make public places safer, we require an automatic system that uses Convolution Neural Networks (CNN) to detect weapons. This system will enhance security, expedite our response to threats, and help in maintaining public places safer by quickly identifying weapons, and averting security concerns.

.

##### 2. Idea / Solution description

For this project we would be working with YOLO (You only look once) a pretrained object detector CNN model which is known for its speed and accuracy. The system will be trained on a dataset containing weapons, and will detect them in real-time, ensuring rapid response to threats.

##### 3. Novelty / Uniqueness

YOLO weapon detection is a novel approach to real-time object detection that is fast and accurate, even in challenging conditions. It can be deployed 24/7, making it a powerful tool for enhancing security and public safety.



#### 4. Social Impact / Customer Satisfaction

YOLO weapon detection enhances public safety and customer satisfaction by deterring crime and protecting lives and property. It could also create job opportunities in system maintenance and oversight, contributing to a safer society

#### 5. Business Model (Revenue Model)

YOLO weapon detection business model relies on equipment sales, maintenance services, and licensing fees. Government partnerships play a crucial role in funding and expansion. By enhancing security, the aim is to boost customer satisfaction and contribute to a safer society.

#### 6. Scalability of the Solution

YOLO weapon detection systems have the capability to integrate with existing security technologies, like video surveillance and access control, resulting in a strong security solution. The system can also be trained to identify new weapon types and operate effectively in challenging environments, thereby enhancing safety and security.

In order to revolutionize real-time weapon identification, our system combines the strength of the YOLO (You Only Look Once) and CNN (Convolutional Neural

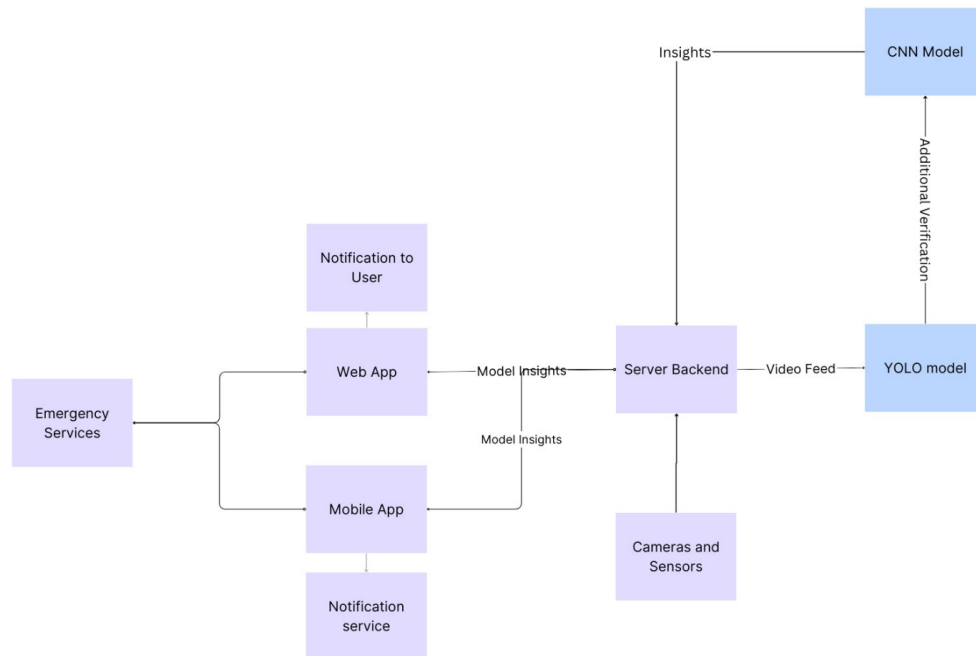
Network) technologies. By implementing this integrated system, we increase the environment's overall safety and security while also improving the accuracy of threat identification. This system does not simply look for weapons; it also makes sure that possible threats are dealt with quickly, minimizing damage and saving lives. The YOLO-CNN framework is built to be adaptable to remain ahead of changing security concerns and to operate with unwavering accuracy.

For real-time threat detection, our design uses YOLO, and CNN adds extra verification and analysis. The following essential parts make up the system:

1. **Camera Network:** Live video feeds are captured by security cameras at various locations.
2. **Real-Time Threat Detection:** YOLO analyses video frames to look for possible weapons.
3. **CNN verification:** The CNN model examines and validates the discovered objects further.
4. **Notification Service:** Both the web app and the mobile app receive instant messages and notifications.
5. **User Interface:** Both the web app and the mobile app offer user-friendly interfaces for logging into the system and getting prompt alerts about risks that have been identified.
6. **Emergency Services Integration:** Users can initiate an emergency services request immediately from the web app and mobile app when a confirmed weapon threat occurs. This capability guarantees a prompt reaction from law enforcement, first responders, or other pertinent authorities to properly resolve security events.

With real-time threat detection and prompt notifications offered through online and mobile apps, this design ensures that the system responds to changing security

needs. By combining the strengths of CNN and YOLO, it represents a paradigm leap in security technology and dramatically improves public safety and security.

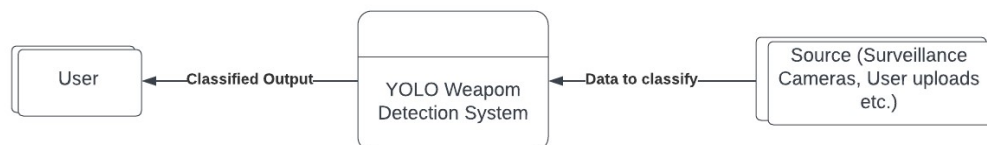


## Solution Architecture Diagram

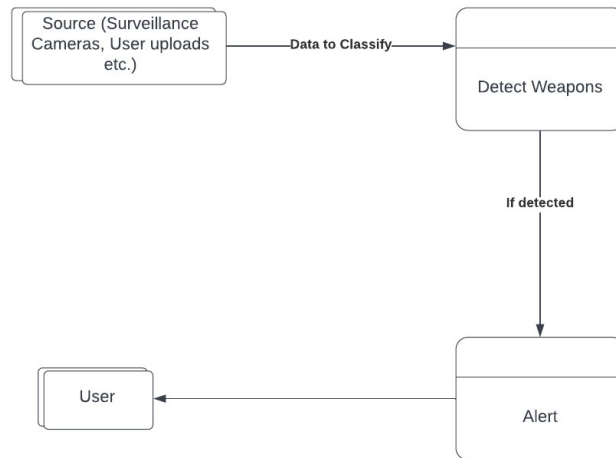
### 5. PRODUCT DESIGN

#### Data Flow Diagrams

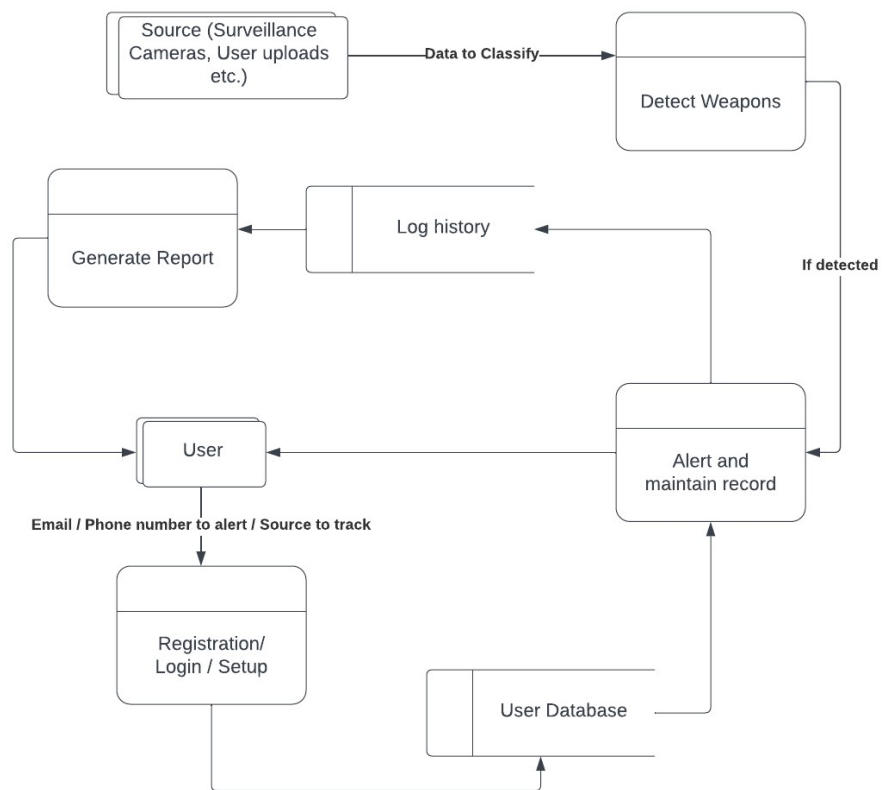
##### DFD Level 0



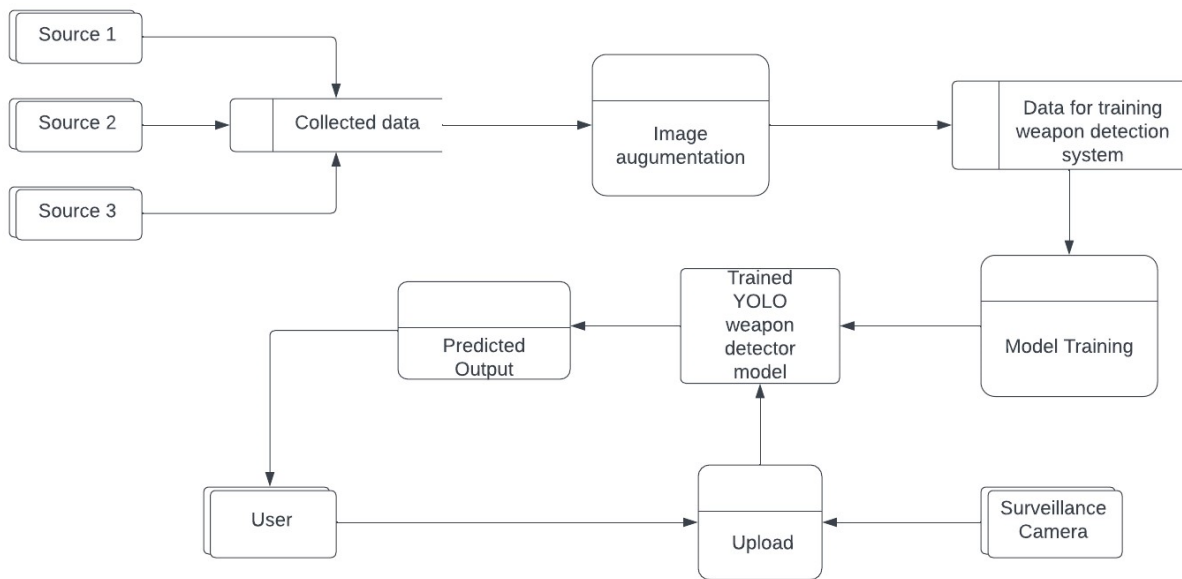
##### DFD Level 1



## DFD Level 2



## Overall Process:



## User Stories

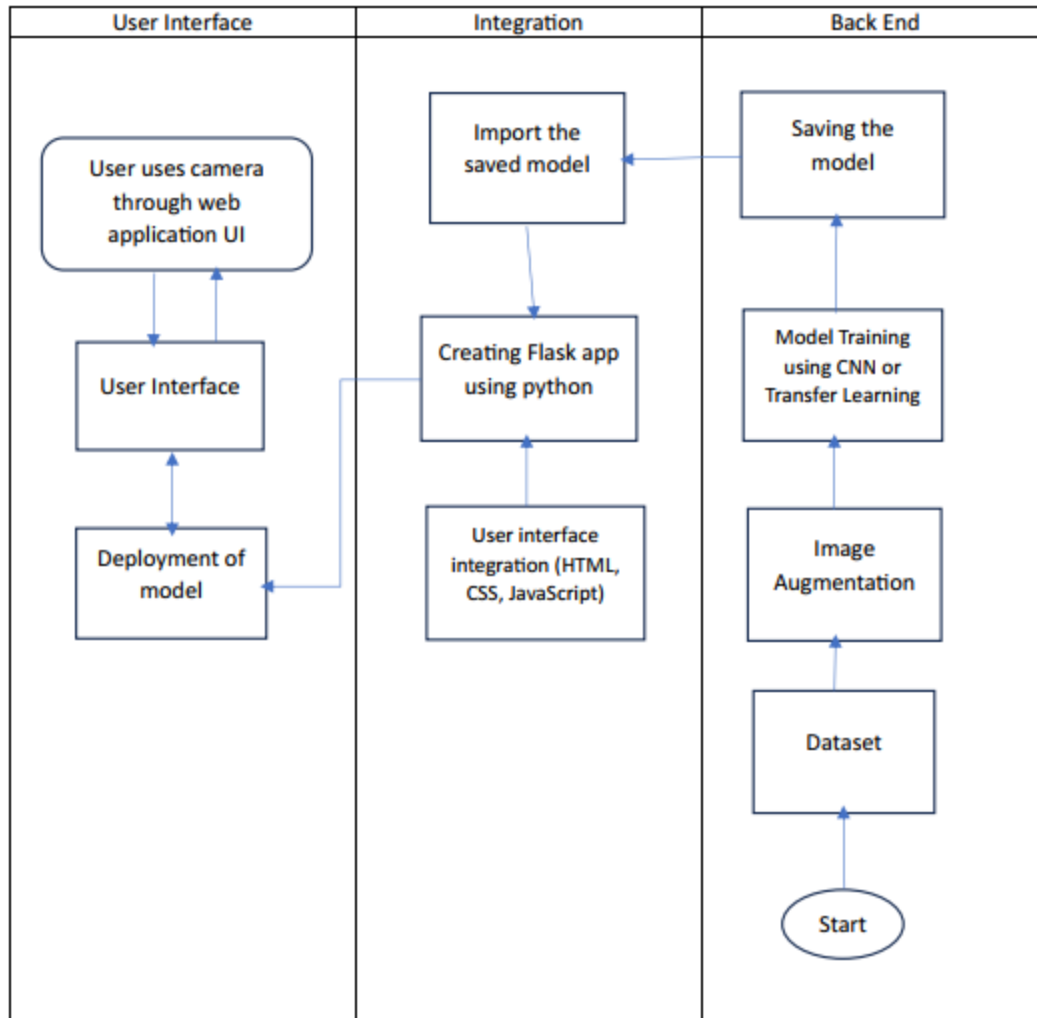
User Type	Functional Requirement (Epic)	User Story Number	User Story / Task	Acceptance criteria	Priority	Release
User (Web user)	Registration	USN – 1	As a user, I can register for the application by entering my email or phone number for alerts, password, and confirming my password.	After completion the user is able to access the application.	High	Sprint 1
User (Web user)	Upload	USN – 2	As a user, I want the capability to upload images for detection.	After completion the user is able to upload images for detection.	High	Sprint 1

User (Web User)	Viewing Detection Results	USN - 3	As a user, I want to view the results of weapon detection performed on uploaded images, including details about detected objects.	Users should have access to view the results of weapon detection for their uploaded images.	High	Sprint 1
User (Web user)	Setup live surveillance	USN – 4	As a user, I want the capability to set up live input from a surveillance camera for real-time monitoring and weapon detection.	After completion the user is able to use live video input for detection.	High	Sprint 2
User	Detect Weapons Accurately	USN – 5	As a user, I expect the YOLO weapon detection system to accurately identify potential weapons, ensuring the safety and security of the premises.	The system should consistently achieve a high accuracy rate in identifying potential weapons.	High	Sprint 2
Security personnel	Alerts	USN – 6	As a security personnel, I expect the YOLO weapon detection system to quickly notify me when potential threats are detected.	Alerts should be sent to security personnel within seconds of detecting a potential threat.	High	Sprint 2

User (Web User)	User Support and Help	USN – 7	As a user, I want access to user support and help resources to get assistance with using the application.	Users have access to FAQs and contact information for customer support.	Medium	Sprint 3
User (Web User)	Feedback and Reporting	USN – 8	As a user, I want the ability to provide feedback and report issues or false positives related to the weapon detection system.	Users are able provide feedback, report concerns, false positives, and suggestions.	Medium	Sprint 3
Security personnel	View alert history	USN – 9	As a security personnel, I want the capability to view the history of alerts and notifications generated by the system.	Security personnel have access to alert history.	Medium	Sprint 3
User (Web User)	Account Recovery	USN – 10	As a user, I want a secure and straightforward account recovery process to regain access to my account in case I forget my password.	User can recover their passwords	Medium	Sprint 3

## 6. PROJECT PLANNING AND SCHEDULING

**Technical Architecture:** The Deliverable shall include the architectural diagram as below and the information as per the table1 & table 2



Guidelines:

1. Include all the processes (As an application logic / Technology Block)
2. Provide infrastructural demarcation (Local / Cloud)
3. Indicate external interfaces (third party API's etc.)
4. Indicate Data Storage components / services
5. Indicate interface to machine learning models (if applicable)



**Table 1: Components and Technologies**

S. No	Component	Description	Technology
1.	User Interface	How user interacts with application e.g.Web U	HTML, CSS, JavaScript / Angular Js /React Js etc.
2.	Application Logic	Logic for a process in the application	Java/Python
3.	Database	Collect the Dataset Based on the Problem Statement	File Manager, MySQL, NoSQL, etc.
4.	File storage/Data	File storage requirements for Storing the dataset	Local System, Google Drive etc.
5.	Framework	Used to Create a web Application, Integrating Frontend and Back End	Python Flask, Django etc.
6.	Deep Learning model	Purpose of Model	CNN, Transfer Learning etc.
7.	Infrastructure (Server/Cloud)	Application Deployment on Local System / Cloud Local Server Configuration: Cloud Server Configuration	Local, Cloud Foundry, Kubernetes, etc.

**Table 2: - Application Characteristics**

S. No	Characteristics	Description	Technology
1.	Open-Source Frameworks	List the open-source frameworks used	Python's Flask
2.	Security Implementations	List all the security / access controls implemented,	Technology used
3.	Scalable Architecture	Justify the scalability of architecture (3 – tier, Micro-services)	Technology used
4.	Availability	Justify the availability of application (e.g., use of load balancers, distributed servers etc.)	Technology used
5.	Performance	Design consideration for the performance of the	Technology used

## **7. CODING AND SOLUTIONING**

Sprint	Functional Requirement (Epic)	User Story Number	User Story / Task	Story Points	Priority	Team Members
Sprint-1	Project setup & Infrastructure	USN-1	As a developer, I aim to configure the development environment with the essential tools and frameworks for initiating the YOLO weapon detection project.	1	High	Girish K
Sprint-1	Data collection	USN-2	As a developer, I aim to procure a diverse dataset of images featuring various types of weapons to be used for training the deep learning model.	2	High	Girish K
Sprint-2	data preprocessing	USN-3	As a developer, I want to preprocess the collected dataset by resizing images, normalizing pixel values, and splitting it into training and validation sets.	3	High	Girish K
Sprint-2	Model Selection	USN-4	As a developer, I aim to investigate and assess various deep learning architectures, such as CNNs, to determine the most suitable model for weapon detection using the YOLO framework.	3	High	Charvi
Sprint-3	Training	USN-5	As a developer, I am responsible for training the chosen deep learning model using the pre-processed dataset and closely monitoring its performance on the validation set.	4	High	Sukanth
Sprint-3	Testing & quality assurance	USN-6	As a developer, I'd like to enhance the model's resilience and accuracy. I would like to incorporate data augmentation methods such as rotation and flipping.	4	Medium	Girish K
Sprint-4	model deployment & Integration	USN-7	As a developer, I'd like to deploy a deep learning model that has been trained for weapon detection as an API or web service. I want to	3	Medium	Charvi

			integrate this model's API into a user-friendly web interface where users can upload images and obtain classification results for the type of garbage in the image.			
Sprint-4	Application testing	USN 8	I plan to perform comprehensive testing of both the model and the web interface, with the aim of discovering and documenting any potential problems or bugs. I will then fine-tune the model's hyperparameters and optimize its performance based on user feedback and the results obtained from the testing.	2	Medium	Charvi

### Project Tracker, Velocity & Burndown Chart: (4 Marks)

Sprint	Total Story Points	Duration	Sprint Start Date	Sprint End Date (Planned)	Story Points Completed (as on Planned End Date)	Sprint Release Date (Actual)
Sprint-1	3	2 Days	16 <sup>th</sup> Oct 2023	17 <sup>th</sup> Oct 2023	22	
Sprint-2	6	4 Days	18 <sup>th</sup> Oct 2023	21 <sup>st</sup> Oct 2023		
Sprint-3	8	3 Days	22 <sup>nd</sup> Oct 2023	24 <sup>th</sup> Oct 2023		
Sprint-4	5	3 Days	25 <sup>th</sup> Oct 2023	27 <sup>th</sup> Oct 2023		

#### **Velocity:**

Imagine we have a 29-days sprint duration, and the velocity of the team is 20 (points per sprint). Let us calculate the team's average velocity (AV) per iteration unit (story points per day)

#### Formula

$$AV = \frac{\text{sprint duration}}{\text{velocity}}$$

#### Average calculation

$$22/10 = 2.2$$

#### **Burndown Chart**

A burndown chart is a graphical representation of work left to do versus time. It is often used in agile software development methodologies such as Scrum. However, burn down charts can be applied to any project containing measurable progress over time.

<https://www.visual-paradigm.com/scrum/scrum-burndown-chart/>

<https://www.atlassian.com/agile/tutorials/burndown-charts>

#### **Reference:**

<https://www.atlassian.com/agile/project-management>

<https://www.atlassian.com/agile/tutorials/how-to-do-scrum-with-jira-software>

<https://www.atlassian.com/agile/tutorials/epics>

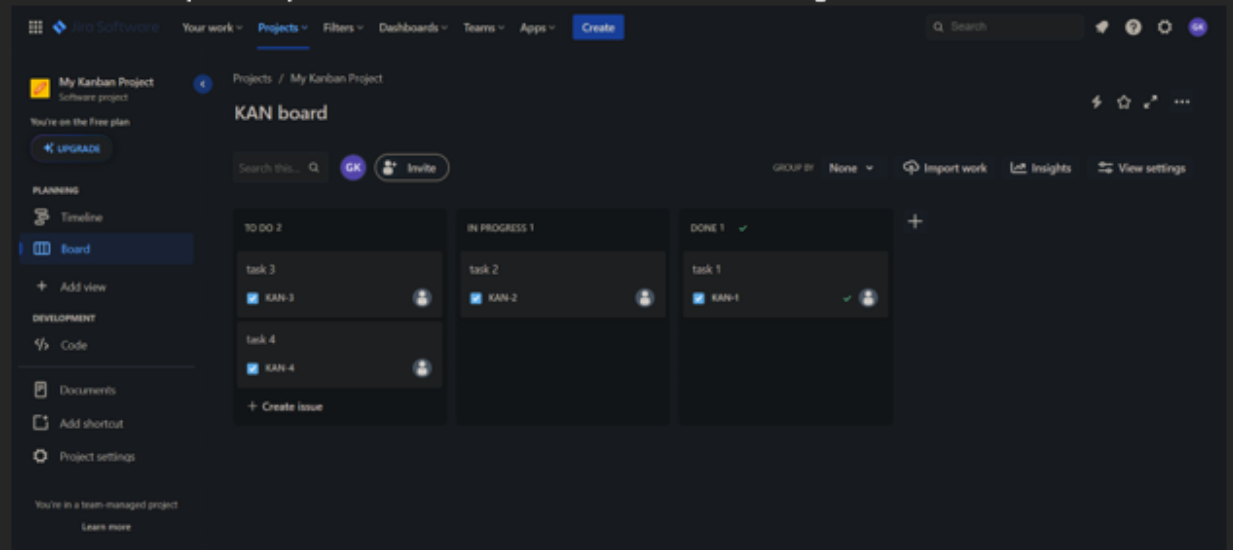
<https://www.atlassian.com/agile/tutorials/sprints>

<https://www.atlassian.com/agile/project-management/estimation>

<https://www.atlassian.com/agile/tutorials/burndown-charts>

## Board section

We have completed sprint 1 and 2. So we can see the remaining tasks on board.



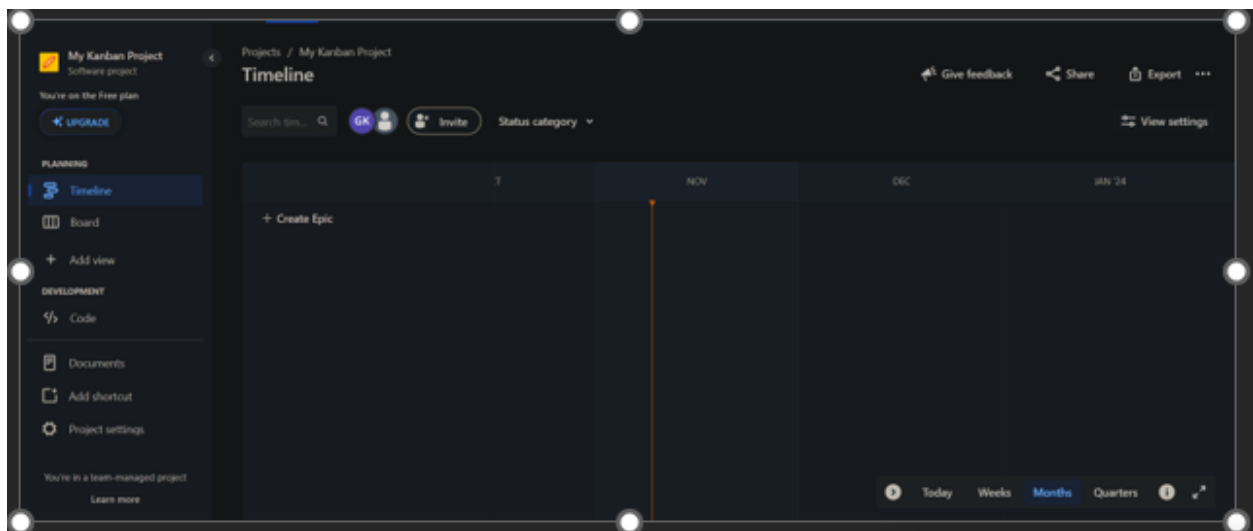
Details of the task were discussing in regularly held google meets with the team where all details of the tasks were assigned and noted

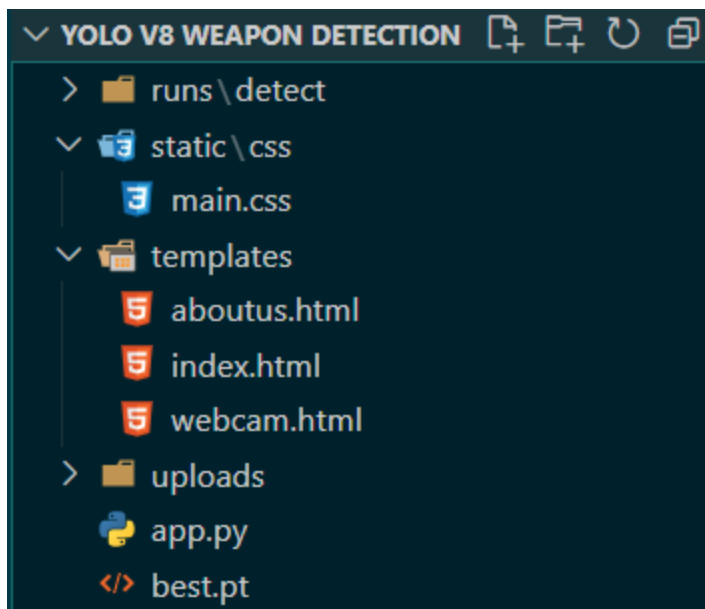
## Backlog section

We have had no significant backlogs during the project

## Timeline

We did not use this tool; we kept the timelines dynamic as the project contained many unknowns.





```
app.py X
app.py > index
1  #Necessary imports
2  from flask import Flask, render_template, request, send_from_directory, send_file, Response
3  import os
4  from urllib.parse import urlparse
5  from ultralytics import YOLO
6  import cv2
7  import time
8  import math
9  import winsound
10
```

Image as source detection

```
31 @app.route("/display/<path:filename>")
32 def display(filename):
33     parsed_url = urlparse(request.url)
34     url_path = parsed_url.path
35     directory, filename = url_path.split("/display/")[1].rsplit("/", 1)
36     input_path = os.path.join(directory, filename).replace("\\", "/")
37     print(input_path)
38     return send_file(input_path, as_attachment=False)
```

```

71     #Image detection Section
72     if file_extension in ("jpg", "png"):
73         img=cv2.imread(filepath)
74         model=YOLO('best.pt')
75         model.predict(source = img, save_txt=True,save=True)
76         folder_path = 'runs/detect'
77         subfolders = [f for f in os.listdir(folder_path) if os.path.isdir(os.path.join(folder_path, f))]
78         latest_subfolder = max(subfolders, key=lambda x: os.path.getctime(os.path.join(folder_path, x)))
79         input_path = folder_path+'/'+latest_subfolder+'/'+image0.jpg"
80         #alarm if weapon detected.
81         with open(os.path.join(folder_path, latest_subfolder, "labels/image0.txt"), "r") as f:
82             for line in f:
83                 if line.strip():
84                     alarm_activated = True
85                     break
86         if alarm_activated:
87             print("Weapon detected")
88             play_alarm()
89         return render_template('index.html', input_path= input_path,video_format=None)

```

## Video as source detection

```

40 @app.route('/video/<path:filename>')
41 def serve_video(filename):
42     video_path = filename
43     def generate_frames():
44         cap = cv2.VideoCapture(video_path)
45         while True:
46             success, frame = cap.read()
47             if not success:
48                 break
49             ret, buffer = cv2.imencode('.jpg', frame)
50             if not ret:
51                 continue
52             frame_bytes = buffer.tobytes()
53             yield (b'--frame\r\n'
54                   + b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')
55             time.sleep(0.1)
56     return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')
57

```

```

91     #Video Section
92     elif file_extension in ("mp4"):
93         model=YOLO('best.pt')
94         model.predict(source = filepath,save_txt=True,save=True)
95         folder_path = 'runs/detect'
96         subfolders = [f for f in os.listdir(folder_path) if os.path.isdir(os.path.join(folder_path, f))]
97         latest_subfolder = max(subfolders, key=lambda x: os.path.getctime(os.path.join(folder_path, x)))
98         input_path = folder_path+'/'+latest_subfolder+'/'+f.filename.replace(".mp4",".avi")
99
100         #alarm if weapon detected.
101         for filename in os.listdir(os.path.join(folder_path, latest_subfolder, "labels/")):
102             if filename.endswith(".txt"):
103                 filepath = os.path.join(folder_path, latest_subfolder, "labels/", filename)
104                 with open(filepath, "r") as f:
105                     # Check for weapon detection
106                     for line in f:
107                         if line.strip():
108                             alarm_activated = True
109                             break
110                 # Play alarm if weapon detected
111                 if alarm_activated:
112                     play_alarm()
113                     print("Weapon detected")
114                     break # Break out of the loop once alarm is activated
115             if not alarm_activated:
116                 # No alarm was activated
117                 print("No weapons detected.")
118         return render_template('index.html', input_path= input_path,video_format="mp4")
119

```

Web Camera as source detection

```

157 @app.route('/webcam_func')
158 def video_feed():
159     return webcam_func()

```

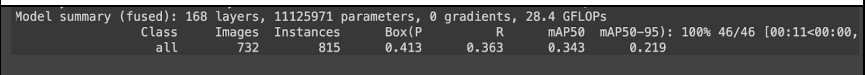
```

121 #WebCam section
122 classNames = ['handgun','knife']
123 @app.route('/webcam_func')
124 def webcam_func():
125     model=YOLO("best.pt")
126     def generate_frames():
127         cap = cv2.VideoCapture(0)
128         while True:
129             success, frame = cap.read()
130             results = model(frame, stream=True)
131             for r in results:
132                 boxes = r.boxes
133                 for box in boxes:
134                     x1, y1, x2, y2 = box.xyxy[0]
135                     x1, y1, x2, y2 = int(x1), int(y1), int(x2), int(y2)
136                     cv2.rectangle(frame, (x1, y1), (x2, y2), (255, 0, 255), 3)
137                     confidence = math.ceil((box.conf[0]*100))/100
138                     print("Confidence --->",confidence)
139                     cls = int(box.cls[0])
140                     print("Class name -->", classNames[cls])
141                     org = [x1, y1]
142                     font = cv2.FONT_HERSHEY_SIMPLEX
143                     fontScale = 1
144                     color = (255, 0, 0)
145                     thickness = 2
146                     cv2.putText(frame, classNames[cls], org, font, fontScale, color, thickness)
147                     play_alarm()
148             if not success:
149                 break
150             ret, buffer = cv2.imencode('.jpg', frame)
151             if not ret:
152                 continue
153             frame_bytes = buffer.tobytes()
154             yield (b'--frame\r\n'
155                 + b'Content-Type: image/jpeg\r\n\r\n' + frame_bytes + b'\r\n')
156             time.sleep(0.1)
157     return Response(generate_frames(), mimetype='multipart/x-mixed-replace; boundary=frame')


```

## 8. PERFORMANCE TESTING

Project team shall fill the following information in the model performance testing template.

S.No.	Parameter	Values	Screenshot
	Model Summary	-	 <pre> Model summary (fused): 168 layers, 11125971 parameters, 0 gradients, 28.4 GFLOPs Class Images Instances Box(P R mAP50 mAP50-95): 100% 46/46 [00:11:00:00, all 732 815 0.413 0.363 0.343 0.219 </pre>



	Accuracy	Training Accuracy -  Validation Accuracy -	<table><tr><td>Epoch</td><td>GPU_mem</td><td>box_loss</td><td>cls_loss</td><td>dfl_loss</td><td>Instances</td><td>Size</td></tr><tr><td>42/50</td><td>0G</td><td>0.6489</td><td>1.2</td><td>1.159</td><td>5</td><td>640: 100% 9/9 [02:08&lt;00:00, 14.28s/it]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% 1/1 [00:08&lt;00:00, 8.71s/it]</td></tr><tr><td>all</td><td>29</td><td>29</td><td>0.685</td><td>0.636</td><td>0.595</td><td>0.44</td></tr><tr><td>Epoch</td><td>GPU_mem</td><td>box_loss</td><td>cls_loss</td><td>dfl_loss</td><td>Instances</td><td>Size</td></tr><tr><td>43/50</td><td>0G</td><td>0.7265</td><td>1.175</td><td>1.196</td><td>5</td><td>640: 100% 9/9 [02:05&lt;00:00, 13.99s/it]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% 1/1 [00:10&lt;00:00, 10.45s/it]</td></tr><tr><td>all</td><td>29</td><td>29</td><td>0.803</td><td>0.64</td><td>0.645</td><td>0.466</td></tr><tr><td>Epoch</td><td>GPU_mem</td><td>box_loss</td><td>cls_loss</td><td>dfl_loss</td><td>Instances</td><td>Size</td></tr><tr><td>44/50</td><td>0G</td><td>0.6954</td><td>1.14</td><td>1.209</td><td>5</td><td>640: 100% 9/9 [02:04&lt;00:00, 13.88s/it]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% 1/1 [00:11&lt;00:00, 11.48s/it]</td></tr><tr><td>all</td><td>29</td><td>29</td><td>0.775</td><td>0.674</td><td>0.655</td><td>0.483</td></tr><tr><td>Epoch</td><td>GPU_mem</td><td>box_loss</td><td>cls_loss</td><td>dfl_loss</td><td>Instances</td><td>Size</td></tr><tr><td>45/50</td><td>0G</td><td>0.7123</td><td>1.09</td><td>1.3</td><td>5</td><td>640: 100% 9/9 [02:02&lt;00:00, 13.57s/it]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% 1/1 [00:11&lt;00:00, 11.13s/it]</td></tr><tr><td>all</td><td>29</td><td>29</td><td>0.726</td><td>0.716</td><td>0.672</td><td>0.477</td></tr><tr><td>Epoch</td><td>GPU_mem</td><td>box_loss</td><td>cls_loss</td><td>dfl_loss</td><td>Instances</td><td>Size</td></tr><tr><td>46/50</td><td>0G</td><td>0.6284</td><td>1.028</td><td>1.191</td><td>5</td><td>640: 100% 9/9 [02:06&lt;00:00, 14.09s/it]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% 1/1 [00:09&lt;00:00, 9.75s/it]</td></tr><tr><td>all</td><td>29</td><td>29</td><td>0.697</td><td>0.705</td><td>0.661</td><td>0.467</td></tr><tr><td>Epoch</td><td>GPU_mem</td><td>box_loss</td><td>cls_loss</td><td>dfl_loss</td><td>Instances</td><td>Size</td></tr><tr><td>47/50</td><td>0G</td><td>0.6283</td><td>1.022</td><td>1.114</td><td>5</td><td>640: 100% 9/9 [02:06&lt;00:00, 14.09s/it]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% 1/1 [00:08&lt;00:00, 8.74s/it]</td></tr><tr><td>all</td><td>29</td><td>29</td><td>0.78</td><td>0.627</td><td>0.653</td><td>0.44</td></tr><tr><td>Epoch</td><td>GPU_mem</td><td>box_loss</td><td>cls_loss</td><td>dfl_loss</td><td>Instances</td><td>Size</td></tr><tr><td>48/50</td><td>0G</td><td>0.6458</td><td>0.9744</td><td>1.136</td><td>5</td><td>640: 100% 9/9 [02:06&lt;00:00, 14.10s/it]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% 1/1 [00:08&lt;00:00, 8.61s/it]</td></tr><tr><td>all</td><td>29</td><td>29</td><td>0.724</td><td>0.707</td><td>0.695</td><td>0.446</td></tr><tr><td>Epoch</td><td>GPU_mem</td><td>box_loss</td><td>cls_loss</td><td>dfl_loss</td><td>Instances</td><td>Size</td></tr><tr><td>49/50</td><td>0G</td><td>0.5825</td><td>0.9801</td><td>1.075</td><td>5</td><td>640: 100% 9/9 [02:06&lt;00:00, 14.00s/it]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% 1/1 [00:09&lt;00:00, 9.65s/it]</td></tr><tr><td>all</td><td>29</td><td>29</td><td>0.736</td><td>0.716</td><td>0.697</td><td>0.46</td></tr><tr><td>Epoch</td><td>GPU_mem</td><td>box_loss</td><td>cls_loss</td><td>dfl_loss</td><td>Instances</td><td>Size</td></tr><tr><td>50/50</td><td>0G</td><td>0.5892</td><td>1.045</td><td>1.116</td><td>5</td><td>640: 100% 9/9 [02:04&lt;00:00, 13.89s/it]</td></tr><tr><td>Class</td><td>Images</td><td>Instances</td><td>Box(P</td><td>R</td><td>mAP50</td><td>mAP50-95): 100% 1/1 [00:10&lt;00:00, 10.92s/it]</td></tr><tr><td>all</td><td>29</td><td>29</td><td>0.742</td><td>0.701</td><td>0.708</td><td>0.469</td></tr></table>	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	42/50	0G	0.6489	1.2	1.159	5	640: 100% 9/9 [02:08<00:00, 14.28s/it]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:08<00:00, 8.71s/it]	all	29	29	0.685	0.636	0.595	0.44	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	43/50	0G	0.7265	1.175	1.196	5	640: 100% 9/9 [02:05<00:00, 13.99s/it]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:10<00:00, 10.45s/it]	all	29	29	0.803	0.64	0.645	0.466	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	44/50	0G	0.6954	1.14	1.209	5	640: 100% 9/9 [02:04<00:00, 13.88s/it]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:11<00:00, 11.48s/it]	all	29	29	0.775	0.674	0.655	0.483	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	45/50	0G	0.7123	1.09	1.3	5	640: 100% 9/9 [02:02<00:00, 13.57s/it]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:11<00:00, 11.13s/it]	all	29	29	0.726	0.716	0.672	0.477	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	46/50	0G	0.6284	1.028	1.191	5	640: 100% 9/9 [02:06<00:00, 14.09s/it]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:09<00:00, 9.75s/it]	all	29	29	0.697	0.705	0.661	0.467	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	47/50	0G	0.6283	1.022	1.114	5	640: 100% 9/9 [02:06<00:00, 14.09s/it]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:08<00:00, 8.74s/it]	all	29	29	0.78	0.627	0.653	0.44	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	48/50	0G	0.6458	0.9744	1.136	5	640: 100% 9/9 [02:06<00:00, 14.10s/it]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:08<00:00, 8.61s/it]	all	29	29	0.724	0.707	0.695	0.446	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	49/50	0G	0.5825	0.9801	1.075	5	640: 100% 9/9 [02:06<00:00, 14.00s/it]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:09<00:00, 9.65s/it]	all	29	29	0.736	0.716	0.697	0.46	Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size	50/50	0G	0.5892	1.045	1.116	5	640: 100% 9/9 [02:04<00:00, 13.89s/it]	Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:10<00:00, 10.92s/it]	all	29	29	0.742	0.701	0.708	0.469
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																																																																																																																																																																																									
42/50	0G	0.6489	1.2	1.159	5	640: 100% 9/9 [02:08<00:00, 14.28s/it]																																																																																																																																																																																																																																																									
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:08<00:00, 8.71s/it]																																																																																																																																																																																																																																																									
all	29	29	0.685	0.636	0.595	0.44																																																																																																																																																																																																																																																									
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																																																																																																																																																																																									
43/50	0G	0.7265	1.175	1.196	5	640: 100% 9/9 [02:05<00:00, 13.99s/it]																																																																																																																																																																																																																																																									
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:10<00:00, 10.45s/it]																																																																																																																																																																																																																																																									
all	29	29	0.803	0.64	0.645	0.466																																																																																																																																																																																																																																																									
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																																																																																																																																																																																									
44/50	0G	0.6954	1.14	1.209	5	640: 100% 9/9 [02:04<00:00, 13.88s/it]																																																																																																																																																																																																																																																									
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:11<00:00, 11.48s/it]																																																																																																																																																																																																																																																									
all	29	29	0.775	0.674	0.655	0.483																																																																																																																																																																																																																																																									
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																																																																																																																																																																																									
45/50	0G	0.7123	1.09	1.3	5	640: 100% 9/9 [02:02<00:00, 13.57s/it]																																																																																																																																																																																																																																																									
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:11<00:00, 11.13s/it]																																																																																																																																																																																																																																																									
all	29	29	0.726	0.716	0.672	0.477																																																																																																																																																																																																																																																									
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																																																																																																																																																																																									
46/50	0G	0.6284	1.028	1.191	5	640: 100% 9/9 [02:06<00:00, 14.09s/it]																																																																																																																																																																																																																																																									
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:09<00:00, 9.75s/it]																																																																																																																																																																																																																																																									
all	29	29	0.697	0.705	0.661	0.467																																																																																																																																																																																																																																																									
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																																																																																																																																																																																									
47/50	0G	0.6283	1.022	1.114	5	640: 100% 9/9 [02:06<00:00, 14.09s/it]																																																																																																																																																																																																																																																									
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:08<00:00, 8.74s/it]																																																																																																																																																																																																																																																									
all	29	29	0.78	0.627	0.653	0.44																																																																																																																																																																																																																																																									
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																																																																																																																																																																																									
48/50	0G	0.6458	0.9744	1.136	5	640: 100% 9/9 [02:06<00:00, 14.10s/it]																																																																																																																																																																																																																																																									
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:08<00:00, 8.61s/it]																																																																																																																																																																																																																																																									
all	29	29	0.724	0.707	0.695	0.446																																																																																																																																																																																																																																																									
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																																																																																																																																																																																									
49/50	0G	0.5825	0.9801	1.075	5	640: 100% 9/9 [02:06<00:00, 14.00s/it]																																																																																																																																																																																																																																																									
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:09<00:00, 9.65s/it]																																																																																																																																																																																																																																																									
all	29	29	0.736	0.716	0.697	0.46																																																																																																																																																																																																																																																									
Epoch	GPU_mem	box_loss	cls_loss	dfl_loss	Instances	Size																																																																																																																																																																																																																																																									
50/50	0G	0.5892	1.045	1.116	5	640: 100% 9/9 [02:04<00:00, 13.89s/it]																																																																																																																																																																																																																																																									
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 1/1 [00:10<00:00, 10.92s/it]																																																																																																																																																																																																																																																									
all	29	29	0.742	0.701	0.708	0.469																																																																																																																																																																																																																																																									
3.	Confidence Score (Only Yolo Projects)	Class Detected - Handgun, Knife  Confidence Score - 0.7+																																																																																																																																																																																																																																																													

## 9. RESULTS

Accuracy:

mAP50	mAP50-95): 100% 1/1 [00:07<00:00, 7.87s/it]
0.655	0.483
0.545	0.441
0.765	0.525

Model Summary:

Model summary (fused): 168 layers, 11125971 parameters, 0 gradients, 28.4 GFLOPs						
Class	Images	Instances	Box(P	R	mAP50	mAP50-95): 100% 46/46 [00:11<00:00,
all	732	815	0.413	0.363	0.343	0.219

Confidence Score:



## 10. ADVANTAGES AND DISADVANTAGES

### Advantages:

Real time detection

Accuracy

### Disadvantages:

Limited to Visible Objects: YOLOv8 primarily detects objects visible in the frame, potentially missing concealed weapons or those obstructed from view.

## 11. CONCLUSION

A web application incorporating YOLOv8 with Flask integration for weapon detection has been developed.

## 12. FUTURE SCOPE

Further enhancing the system with a login section connected to a database will enable notifying users about weapon detection through email or messages.

## 13. APPENDIX

Dataset Link

[https://drive.google.com/drive/folders/1fv9HCfP9DvQwhhcypo08hbf2xdZI6LL3?usp=share\\_link](https://drive.google.com/drive/folders/1fv9HCfP9DvQwhhcypo08hbf2xdZI6LL3?usp=share_link)

Github and Project Demo Link:

Github: <https://github.com/smartinternz02/Sl-GuidedProject-591271-1697127617>

Project Demo: <https://drive.google.com/file/d/1i4TG-R2lqafYANCRmliO1-TIMjGj9LYs/view>